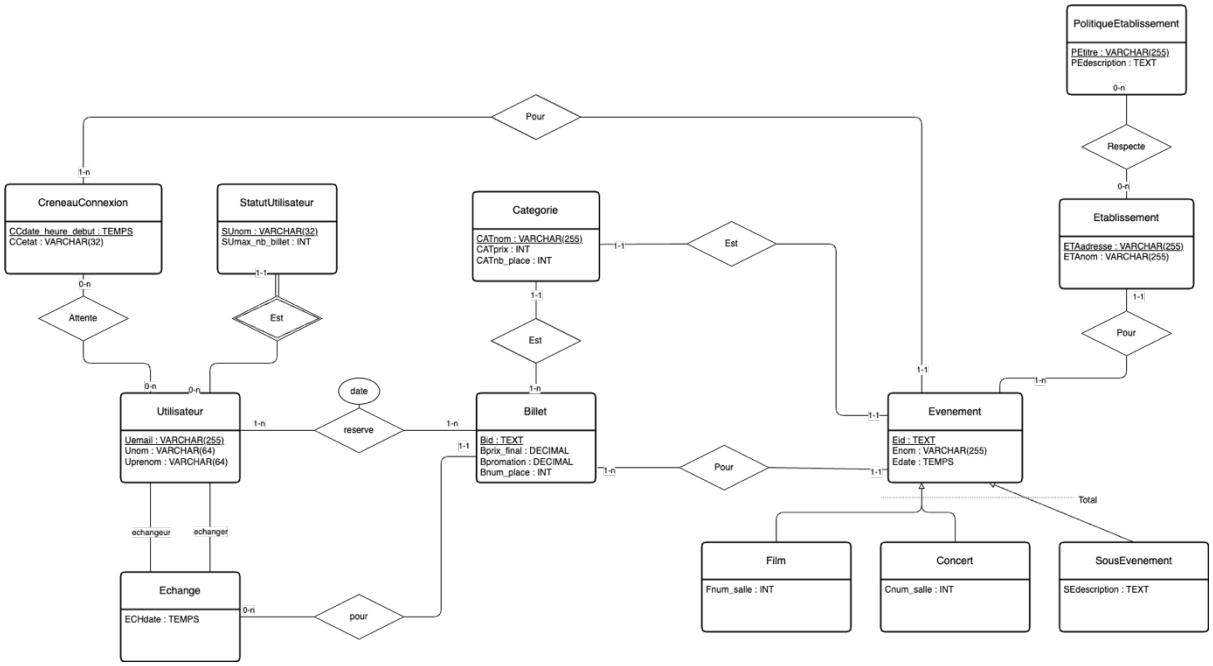
Introduction

Ce projet a pour objectif la conception et la réalisation d'un système de réservation de billets complexe, inspiré du Grand Rex et des événements de l'Accor Arena. Le système intègre des règles avancées de gestion des réservations, la création de créneaux de connexion pour éviter la saturation, ainsi qu'un contrôle dynamique des droits des utilisateurs selon leur statut. Ce rapport présente la modélisation conceptuelle, les contraintes, les triggers, les scénarios d'utilisation, ainsi que les choix d'implémentation.

Diagramme v2



Au cours de l'évolution du projet, plusieurs ajustements ont été effectuer sur la modélisation conceptuelle. Ces modifications sont les suivantes :

Suppressions :

- Avantage : Jugée redondant ou peut utile dans la cadre de notre cas d'usage.

- HistoriqueTransaction : Sa logique a été intégrée dans Echange, Reservation et Billet.

- Seance : Fusionnée avec Evenement pour éviter une complexité inutile.

- PreReservation : Remplacer par une relation Reservation.

Modifications :

- Association Echange : Inclure les clé primaire des utilisateur (émetteur, destinataire) et du billet, avec ajout des attributs Ejour et Eheure pour tracer la transaction.

- Evenement specialises :

    - Film

    - Concert

    - Sous-Evenement

    Ces sous-types dérivent de l'entité Evenement, conforme au besoin d'hériter des comportement et attributs spécifique.

Modélisation Conceptuelle

Entités principales :

1. Utilisateur (Uemail, Unom, Uprenom, Ustatut, Ususpect, Uconnecte)

2. CreneauConnexion (CCjour_debut, CCheure_debut, CCetat, CCmax_connexions)

3. Billet (Bid, Bjour_achat, Bheure_acha, Bprix_initial, Bprix_achat, Bpromotion, Bdiponibilite, Bnum_place)

4. PolitiqueEtablissement(*PEtitre*, description)

5. Etablissement (ETAadresse, ETAnom)

6. Evenement(Enom_complet, Enom, Ejour, Eheure, Enum_salle, Edescription, Etype)

7. Echange(Uemail_emetteur, Uemail_destinataire, Bid, Ejour, Eheure)

8. Achat(Uemail, Bid, Ajour, Aheure)

9. Reservation(Umail, Bid, Rjour_debut, Rheure_debut, Rstatut)

10. CategorieBillet (CATnom, Bid)

11. CategorieEvenement(CATnom, Eid)

Contraintes :

Notre modèle inclut des contraintes pour garantir la cohérence des données. Elles sont classées en contraintes simples (CHECK, UNIQUE) et complexes (TRIGGER).

Contrainte	Type	Implémentation SQL
Un utilisateur doit avoir un email unique	UNIQUE	Email VARCHAR (255) UNIQUE
L'état des créneaux de connexion sont parmi : **- Ouvert** **- Ferme** **- En attente**	CHECK	CHECK (CCetat IN ('Ouvert', 'Ferme', 'En attente'))
Le jour et heure de créneau de connexion ne doivent pas être négatif	CHECK	CHECK (CCjour_debut >= 0 AND CCheure_debut >= 0)
Prix d'achat de Billet non-inférieur ou égal 0	CHECK	CHECK (Bprix_achat > 0)
Prix initial de Billet non-inférieur ou égal 0	CHECK	CHECK (Bprix_initial > 0)
Promotion du billet est un pourcentage	CHECK	CHECK (Bpromotion >= 0 AND Bpromotion <= 100)
Le prix par categorie ne peuvent pas etre negative	CHECK	CHECK (CATprix > 0)
Le numero de place du categorie ne peut pas etre negative	CHECK	CHECK (CATnb_place > 0)
Le type de evenement est parmi : **- Concert** **- Film** **- SousEvenement**	CHECK	CHECK (Etype IN ('Concert', 'SousEvenement', 'Film'))
Le statut de reservation est parmi : **- Pre-reserve** **- Preserve** **- Annule** **- Confirme**	CHECK	CHECK (Rstatut IN ('Pre-reserve', 'Reserve', 'Annule', 'Confirme'))
Le nombre maximum de billet que un utilisateur peut achete pour un evenement, depend du statut de l'utilisateur	TRIGGER	calcul_nb_max_billets_trigger
Creation d'un identifiant de billet coherent et unique	TRIGGER	creer_billets_trigger
Determier la categorie d'un billet en fonction de l'identifient du billet	TRIGGER	ajout_categorie_billet_trigger
Determier la categorie d'un evenement en fonction de l'identifient de l'evenement	TRIGGER	lien_categorie_evenement
La suppression de evenement doit supprimer les billets assossier	TRIGGER	check_after_time
L'existance de l'utilisateur de destinataire	TRIGGER	Verif_utilisateur_destinataire_trigger

Scenarios d’utilisation et démonstration

Les scenario suivant illustrent les principales interactions entre les utilisateurs et le system. Ils mettent en œuvre les fonctions, triggers et règles de gestion implémentés. Chaque scenario présent sont objectif, les tables impactées, ainsi que les comportements attendus.

Insertion d’un utilisateur

· Objectif : Ajouter un nouveau compte utilisateur.

· Tables impliquées : Utilisateur

· Comportement : Insertion uniquement si l’email est unique. Possibilité d’ajouter une vérification du mot de passe (non implémentée ici).

· Résultat attendu : Nouvel utilisateur ajouté ; en cas d’email en doublon → rejet par contrainte UNIQUE.

Connexion d’un utilisateur

· Objectif : Connecter un utilisateur et enregistrer son créneau.

· Tables impliquées : Utilisateur, CreneauConnexion

· Comportement : Mise à jour du champ Uconnecte à true si un créneau est ouvert et encore disponible.

· Résultat attendu : Statut connecté mis à jour, nombre de connexions décrémenté dans le créneau.

Déconnexion d’un utilisateur

· Objectif : Gérer la déconnexion et libérer une place dans le créneau.

· Tables impliquées : Utilisateur, CreneauConnexion

· Comportement : Statut Uconnecte mis à false, incrément du compteur de connexions restantes dans le créneau.

· Résultat attendu : Libération correcte du créneau.

Insertion d’un événement

· Objectif : Ajouter un nouvel événement et l’associer à un établissement.

· Tables impliquées : Evenement, Etablissement, EvenementEtablissement

· Comportement : Création d’un événement et enregistrement de son lien avec l’établissement concerné.

· Résultat attendu : Nouvel événement disponible dans le système.

Pré-réservation d’un billet

· Objectif : Réserver temporairement un billet disponible.

· Tables impliquées : Billet, Réservation

· Comportement : Vérifie la disponibilité du billet. Crée une réservation avec le statut Pre-reserve et met à jour le billet.

· Résultat attendu : Billet marqué comme pré-réservé.

Confirmation de réservation

· Objectif : Valider une pré-réservation.

· Tables impliquées : Réservation, Billet

· Comportement : Mise à jour du statut de la réservation à Confirme et ajustement du prix d'achat dans la table Billet.

· Résultat attendu : Réservation confirmée, billet facturé.

Réservation directe (hors pré-réservation)

· Objectif : Réserver immédiatement un billet.

· Tables impliquées : Billet, Réservation

· Comportement : Vérifie la disponibilité, crée une entrée Réservation avec le statut Reserve et met à jour le billet.

· Résultat attendu : Réservation instantanée validée.

Annulation d'une réservation

· Objectif : Annuler une réservation et remettre le billet en circulation.

· Tables impliquées : Réservation, Billet

· Comportement : Statut de la réservation mis à Annule, et disponibilité du billet remise à true.

· Résultat attendu : Billet de nouveau disponible à la réservation

Détection d'utilisateurs suspects

· Objectif : Identifier les utilisateurs effectuant un nombre anormal de réservations.

· Tables impliquées : Utilisateur, Réservation

· Comportement : Requête identifiant les utilisateurs ayant un nombre de réservations ≥ seuil défini.

· Résultat attendu : Liste des utilisateurs marqués comme suspects (Ususpect = true).

Gestion dynamique des créneaux de connexion

· Objectif : Déterminer si un utilisateur peut accéder à la plateforme.

· Tables impliquées : CreneauConnexion, Utilisateur

· Comportement : Fonction qui teste si le créneau est Ouvert et non saturé.

· Résultat attendu : Accès autorisé ou refusé.

Insertion d'un événement privé

· Objectif : Créer un événement réservé à certains utilisateurs via créneaux spécifiques.

· Tables impliquées : Evenement, CreneauConnexion, CreneauConnexionEvenement

· Comportement : Insère un événement et le lie à un créneau spécifique pour filtrer l'accès.

· Résultat attendu : Événement accessible uniquement à travers un créneau préconfiguré.

Initier échange de billet

· Objectif : Créer une annonce d'échange d'un utilisateur sur un billet qu'il possède.

· Table impliquées : Echange

· Comportement : Insère un échange, et le lie à un utilisateur d'émetteur.

· Résultat attendu : Voir les utilisateurs qui veulent faire des échanges.

Effectuer l'échange de billet

· Objectif : Confirmer un échange entre l'émetteur et le destinataire.

· Table impliquées : Echange, Reservation, Billet

· Comportement : Changer le prix d'achat du billet.

· Résultat attendu : Garder une trace de la transaction.

Transaction de réservation

· Objectif : Réaliser en une seule opération la pré-réservation, la confirmation et l'application d'une éventuel promotion pour un billet donne.

· Table impliquées : Reservation, Billet

· Comportement : Si le billet est disponible, il est d'abord pré-réservé, puis confirme avec le prix mis à jour.

· Résultat attendu : Le billet est associé à l'utilisateur avec le statut Confirme, le prix mis à jour selon la promotion.

Index

Afin d'améliorer les performances du système, des index ont été créés sur certaines colonnes fréquemment sollicitées dans les requêtes. Ces index permettent de réduire le temps de réponse lors de l'accès ou du filtrage des données sur des champs critiques.

Table	Colonne indexée	Justification
Reservation	Rjour_debut, Rheure_debut	Ces champs sont utilisés pour vérifier les disponibilités dans les créneaux horaires lors des réservations.
CreneauConnexion	CCjour_debut, CCheure_debut	Permet de retrouver efficacement les créneaux ouverts ou saturés à une date/heure donnée.
Evenement	Enom_complet, Ejour,Eheure	Optimise la recherche des événements par nom et horaire, utile dans l'affichage des événements futurs ou filtrés.
Billet	Bdisponible	Accélère la recherche des billets disponibles pour une réservation ou une revente.
Reservation	Uemail, Bid	Permet de savoir rapidement si un utilisateur a déjà réservé un billet donné, évitant les doublons.